SKILLS MATTER

GRAILS & DOJO

2007

GRAILS eXCHANGE

http://www.grails-exchange.com | http://www.grails.org | http://skillsmatter.com

About Sven

- Using Groovy & Grails since Aug 2006
- Grails Podcast, Groovy Series
- Working at Yahoo!, Inc

- Find out yourself www.svenhaiges.de

Goals

- You know how to get started!
- You know how AJAX is supported by Grails
  - Tags, Libraries, Abstraction Layer
- You know why Groovy & Grails support your AJAX development, using Dojo
  - Controllers & Actions, render(), Builders, Templates
  - Tipps, gotchas, further resources

Getting Started

Getting started...

- Install Grails – done!
- Install Dojo
  - Best
    - download from dojotoolkit.org
    - Place in web-app/js
    - Or use the CDN version of dojo, not installation on your own server!

grails install-dojo installs dojo 0.4.3 !

How to add Dojo to your pages

- 0.4.3:
  - Load the dojo core library in your pages
    <g:javascript library="dojo" />

- Want to use dojo 0.9? Just use CDN:

```
<script type="text/javascript"
        djConfig="isDebug: true"
        src="http://o.aolcdn.com/dojo/0.9.0/dojo/dojo.xd.js"></script>
<script type="text/javascript">


        dojo.addOnLoad(function(){
                alert('loaded, all good');
        });
</script>
```

Understanding Grails & AJAX

- What a surprise, Grails can serve JavaScript, too!
- Grails supports AJAX with special tags that provide basic AJAX support for these libraries: prototype, Yahoo! UI, Dojo
  - Abstraction Layer
- Dojo currently has to be installed separately, Y UI and Prototype come with.

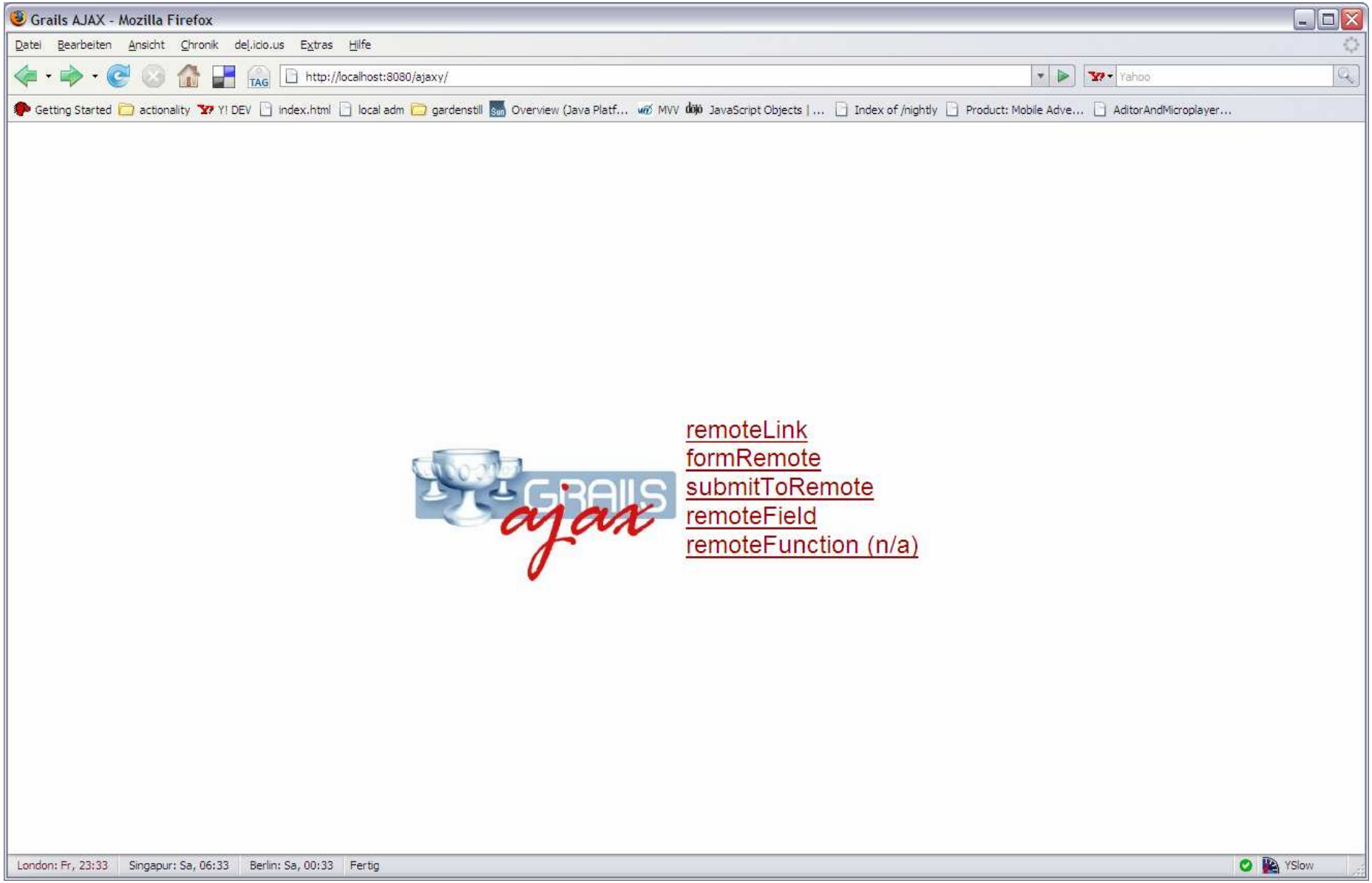
Dojo is more than AJAX

- Many say AJAX but really mean AJAX + Widgets.
- Dojo is very strong in Widgets.
- Besides this, you get way more: package system, animations, utilities, Shrinksafe
- But: you have to code this in your own code, in javascript. No tags can help you here.
  - We will focus on the core AJAX functionality for this presentation.

Learning Path

- Get familiar with Grails
- Use some Grails AJAX Tags in your pages, see what source code they generate
- Experiment with your own Javascript / Dojo code
- Really understand JavaScript & CSS

Grails AJAX Tags



remoteLink

- Creates an <a> Link, submits the request via AJAX and is capable of replacing a div upon response. Several onXXX Methods to customize the call.

```
<div id="message"></div>
<g:remoteLink
        action="remoteLinkCallId"
        id="1"
        update="message">
                AJAX Call, id=1
</g:remoteLink>
```

```
def remoteLinkCallId = {
        log.debug("${actionName} Action called with ${params.id}")
        render "You called ${actionName} in ${controllerName} with
         ${params.id}"
}
```

formRemote

- Creates a <form> that will submit all input fields via AJAX.

```
<g:formRemote
        url="[action:'formRemoteCall']"
        name="form2"
        update="message"
        onLoading="toggleSpinner(true)"
        onLoaded="toggleSpinner(false)" >
                User: <input name="user" type="text"></input>
                <input type="submit" value="formRemote Call"></input>
</g:formRemote>
```

```
def formRemoteCall = {
        log.debug("${actionName} Action called with ${params.user}")
        render "You called ${actionName} in ${controllerName} with
         ${params.user}"
}
```

submitToRemote

- Same as formRemote, just the submit logic is executed on a specific submit button, not all submit buttons.

```
<g:javascript library="dojo" />
<g:javascript>
        dojo.require("dojo.io.IframeIO");
</g:javascript>

<g:form url="[action:'submitToRemoteCall']" id="form2"
        enctype="multipart/form-data">
                File: <input name="someFile" type="file"></input>
                <g:submitToRemote
                value="Submit Upload"
                name="form2"
                action="submitToRemoteUpload"
                update="[success:'message',failure:'error']" />
</g:form>
```

submitToRemote

- Form uploads require us to send the response in an HTML textarea field!

```
def submitToRemoteUpload =
{
        def f = request.getFile('someFile')

        if(f.empty) {
                render "No file!"
        } else {
def fileName = f.getOriginalFilename()
render(text:"<html><body><textarea>You called ${actionName} in
${controllerName} with file ${fileName}</textarea></body></html>",
contentType:"text/html", encoding:"UTF-8")
        }
}
```

remoteField

- Creates an <input> field that submits itself automatically

```
<g:remoteField before="if (this.value.length < 3) return false;"
        action="quickSearch" update="tableContent" name="search"
        paramName="search"/>
<span id="spinner" style="display:none;">
<img src="${createLinkTo(dir:'images',file:'spinner_mac.gif')}"
alt="Spinner" />
</span>
```

```
def quickSearch = {
        def devices = Device.findAllByModelLike("%${params.search}%",
        [max:20, sort:"model", order:'asc'])

        render(template:'tableContent', model:[deviceList:devices])
}
```

remoteField

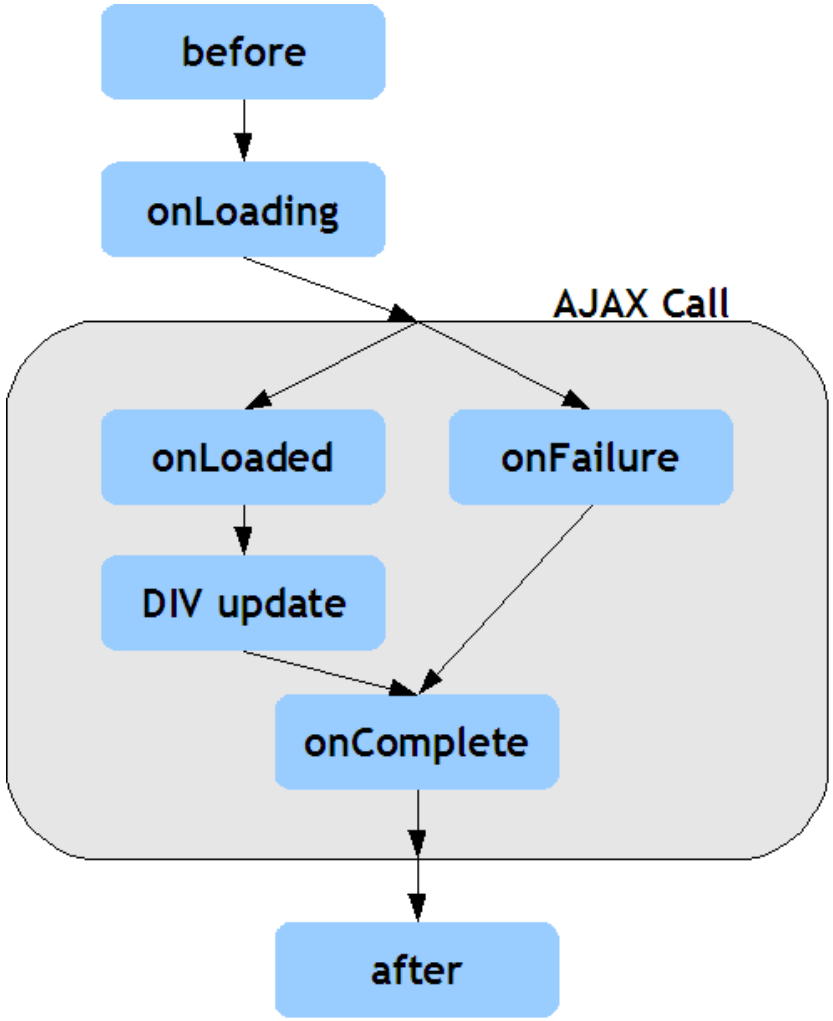
list.gsp

```
<table>
      <thead>
       <tr>
    ...
       </tr>
       </thead>
       <tbody id="tableContent">
       <g:each in="${deviceList}" var="device">
             ...
       </g:each>
       </tbody>
</table>
```

_tableContent.gsp

```
<g:each in="${deviceList}" var="device">
    <tr>
            <td>${device.id}</td>
            ...
    </tr>
</g:each>
```

onXXX Methods



Grails AJAX Tags

- Basic AJAX functionality, good to get started or for very easy use cases
- More complex stuff: master dojo
  - E.g. define the content you send yourself
  - 0.9
  - Complex UI changes

Grails AJAX Tags do not provide

- Support for Dijit Widgets
- … but Grails is the ideal server part for many data-hungry widgets – next section.

Grails AJAX support

Render() is your best friend

- Whether you render pure text, html or JSON/XML formatted text, render() does it all.
- Creating simple responses is quick and easy:

```
render "This is easy.“
render "{font:{id:10, name:'Arial'}}"
```

- If you got larger, more complex responses, use a template to keep it clean:

```
render(template:'podcastList', model:[podcasts:Podcast.findAll()])
```

Rendering JSON

- JSON is faster than XML, it should be AJAJ
- XML is evil, don't use it
- Really, guess what's faster:

<?xml version="1.0" encoding="ISO-8859-1"?><linked-hash-map> <entry>
<string>identifier</string> <string>id</string> </entry> <entry>
<string>items</string> <list> <linked-hash-map> <entry> <string>id</string>
<long>1</long> </entry> <entry> <string>url</string>
<string>http://url1/rss</string> </entry> </linked-hash-map> <linked-hash-
map> <entry> <string>id</string> <long>2</long> </entry> <entry>
<string>url</string> <string>http://url2/rss</string> </entry> </linked-hash-
map> <linked-hash-map> <entry> <string>id</string> <long>3</long> </entry>
<entry> <string>url</string> <str
</linked-hash-map> <linked-hash-m
<long>4</long> </entry> <entry> <
<string>http://url4/rss</string>
map> <entry> <string>id</string>
<string>url</string> <string>htt
map> <linked-hash-map> <entry> <s
<entry> <string>url</string> <str
</linked-hash-map> <linked-hash-m
<long>7</long> </entry> <entry> <string>url</string>
<string>http://url7/rss</string> </entry> </linked-hash-map> <linked-hash-
map> <entry> <string>id</string> <long>8</long> </entry> <entry>
<string>url</string> <string>http://url8/rss</string> </entry> </linked-hash-
map> <linked-hash-map> <entry> <string>id</string> <long>9</long> </entry>
<entry> <string>url</string> <string>http://url9/rss</string> </entry>
</linked-hash-map> <linked-hash-map> <entry> <string>id</string>
<long>10</long> </entry> <entry> <string>url</string>
<string>http://url10/rss</string> </entry> </linked-hash-map> </list>
</entry> </linked-hash-map>

XML or JSON?

{"identifier":"id","items":[{"id":1,"url":
"http://url1/rss"},{"id":2,"url":"http://u
rl2/rss"},{"id":3,"url":"http://url3/rss"}
l4/rss"},{"id":5,
,{"id":6,"url":"ht
","url":"http://url7
/rss"},{"id":8,"url":"http://url8/rss"},{"
id":9,"url":"http://url9/rss"},{"id":10,"
url":"http://url10/rss"}]}

JSON is easy to read, too

```
{
   "identifier":"id",
   "items":
      [
      {
         "id":1,
         "url":"http://url1/rss"
      },
      {
         "id":2,
         "url":"http://url2/rss"
      },
      {
         "id":3,
         "url":"http://url3/rss"
      },
      {
         "id":4,
         "url":"http://url4/rss"
      }
      ]
}
```

JSON is easy to create, too

```
render(builder:'json')
{
 element(imagePath:"/app/myimage.png")
}
```

```
{"element":{"imagePath":"/app/myimage.png"}}
```

- Then work with it in you app:

```
dojo.io.iframe.send(
{
        url: "/some/url",
        form: dojo.byId('formImage'),
        handleAs: "json",
        method: "POST",
        handle: function(response, ioArgs)
        {
                var fileName = response.elementImage.fileName;
        }
});
```

Using converter & Render

- Converts domain objects to JSON

```
import grails.converters.JSON
def podcastJSON = {
    def podcast = Podcast.get(1)
    render podcast as JSON
}
```

```
{
        "id":1,
        "class":"Podcast",
        "author":"Author1",
        "feedURL":"http://url1/rss"
}
```

Special Dojo Widgets & Grails

Data-hungry widgets

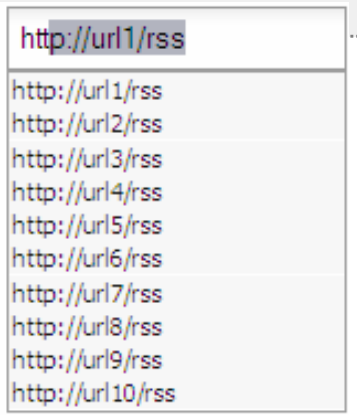
- ComboBox
  - Provides a list of acceptable values but user can still enter his own value. Has a value, just like a textbox.
- FilteringSelect
  - Much like ComboBox, but has label/value like a select. Will set 'identifier' as value of selected label.
- Tree

Dojo ComboBox & Grails

- GSP

```
<div dojoType="dojo.data.ItemFileReadStore" jsId="stateStore"
url="<g:createLink controller="widget" action="comboboxConverterData“ />">
</div>

<input    dojoType="dijit.form.ComboBox"
          store="stateStore"
          hasDownArrow="false"
          value=""
          searchAttr="url"
          name=“feed"
          onChange="setValue" />
```



Dojo ComboBox & Grails

- Grails Action

```
def comboboxConverterData = {
        def items = []
        def podcasts = Podcast.findAll();
        podcasts.each { podcast ->
                items << [id:podcast.id, url:podcast.feedURL]
        }

        def json = [identifier:"id", items: items]

        render json as JSON
}
```

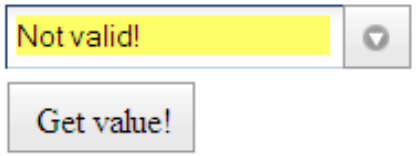
- Using the Grails Converter simplifies the creation of the expected json structure.

Dojo FilterSelect & Grails

- GSP

```
<div dojoType="dojo.data.ItemFileReadStore" jsId="feedStore"
url="<g:createLink controller="widget" action="filterSelectData" />">
</div>

<input dojoType="dijit.form.FilteringSelect"
        id="chooser"
        store="feedStore"
        searchAttr="url"
        name="feed"
        autocomplete="true"
        pageSize="5"
 />
```



Dojo FilterSelect & Grails

- Grails Action – the same…but!

```
def filterSelectData = {
        def items = []
        def podcasts = Podcast.findAll();
        podcasts.each { podcast ->
                items << [id:podcast.id, url:podcast.feedURL]
        }

        def json = [identifier:"id", items: items]
        render json as JSON
}
```

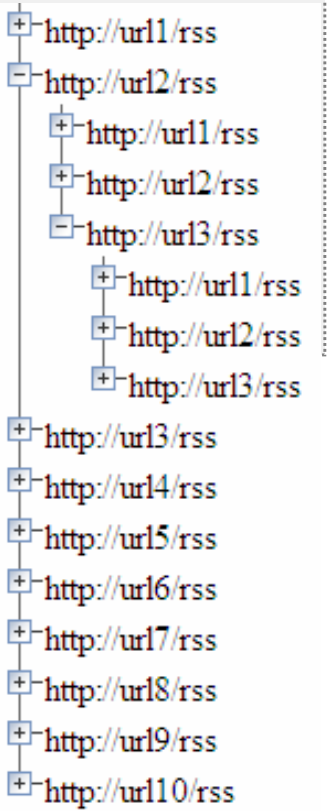
- Once the user has chosen a label from the list, the identifier (here: podcast.id value) will be used as the value of the widget.

Dojo Tree & Grails

- GSP

```
<div dojoType="dojo.data.ItemFileReadStore" jsId="treeStore"
url="<g:createLink controller="widget" action="treeData" />">
</div>

<div     dojoType="dijit.Tree"
         store="treeStore"
         childrenAttr="children"
         labelAttr="url">
</div>
```



Dojo Tree & Grails

- Grails Action

```
def treeData = {
def items = []
def children = []

(1..3).each {
        children  << [_reference:"${it}"]
}

def podcasts = Podcast.findAll();
podcasts.each { podcast ->
items << [id:podcast.id, url:podcast.feedURL, children:children]
}

def json = [identifier:"id", label:'url', items: items]

render json as JSON
}
```

Tipps & Resources

Resources

- Download Dojo, check the test directories for tons of examples
- Dojo API Tool soon available for 0.9
  - http://www.dojotoolkit.org/api
- Converters Plugin
  - http://www.grails.org/Converters+Plugin
- Dynamic Controller Methods, incl. render()
  - grails.org/Controller+Dynamic+Methods

Resources

- JSON Formatter
  - http://www.curiousconcept.com/jsonformatter
- Blogs blogs blogs
- Grails Podcast
  - http://hansamann.podspot.de/rss

Thx

Y IM/SKYPE hansamann